UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Technology
Robotics and Computer Engineering Curriculum

Asif Sattar

Human detection and distance estimation with monocular camera using YOLOv3 neural network

Master's Thesis (30 ECTS)

Supervisors:
Karl Kruusamäe, PhD
Lauri Tammeveski, MSc

Tartu 2019

Human detection and distance estimation with monocular camera using YOLOv3 neural network

Making machines perceive environment better or at least as well as humans would be beneficial in lots of domains. Different sensors aid in this, most widely used of which is monocular camera. Object detection is a major part of environment perception and its accuracy has greatly improved in the last few years thanks to advanced machine learning methods called convolutional neural networks (CNN) that are trained on many labelled images. Monocular camera image contains two dimensional information, but contains no depth information of the scene. On the other hand, depth information of objects is important in a lot of areas related to autonomous driving, e.g. working next to an automated machine, pedestrian crossing a road in front of an autonomous vehicle, etc.

This thesis presents an approach to detect humans and to predict their distance from RGB camera for off-road autonomous driving. This is done by improving YOLO (You Only Look Once) v3[1], a state-of-the-art object detection CNN. Outside of this thesis, an off-road scene depicting a snowy forest with humans in different body poses was simulated using AirSim and Unreal Engine. Data for training YOLOv3 neural network was extracted from there using custom scripts. Also, network was modified to not only predict humans and their bounding boxes, but also their distance from camera. RMSE of 2.99m for objects with distances up to 50m was achieved, while maintaining similar detection accuracy to the original network. Comparable methods using two neural networks and a LASSO model gave 4.26m (in an alternative dataset) and 4.79m (with dataset used is this work) RMSE respectively, showing a huge improvement over the baselines. Future work includes experiments with real-world data to see if the proposed approach generalizes to other environments.

Keywords:
Monocular camera, object detection, depth estimation, CNN, off-road, YOLOv3, simulation

CERCS: T111 Imaging, image processing
P176 Artificial Intelligence

Inimeste tuvastamine ning kauguse hindamine kasutades kaamerat ning YOLOv3 tehisnärvivõrku

Inimestega vähemalt samal tasemel keskkonnast aru saamine masinate poolt oleks kasulik paljudes domeenides. Mitmed erinevad sensored aitavad selle ülesande juures, enim on kasutatud kaameraid. Objektide tuvastamine on tähtis osa keskkonnast aru saamisel. Selle täpsus on viimasel ajal palju paranenud tänu arenenud masinõppe meetoditele nimega konvolutsioonilised närvivõrgud (CNN), mida treenitakse kasutades märgendatud kaamerapilte. Monokulaarkaamerapilt sisaldab 2D infot, kuid ei sisalda sügavusinfot. Teisalt, sügavusinfo on tähtis näiteks isesõitvate autode domeenis. Inimeste ohutus tuleb tagada näiteks töötades autonoomsete masinate läheduses või kui jalakäija ületab teed autonoomse sõiduki eest.

Antud töös uuritakse võimalust, kuidas tuvastada inimesi ning hinnata nende kaugusi samaaegselt, kasutades RGB kaamerat, eesmärgiga kasutada seda autonoomseks sõitmiseks maastikul. Selleks täiustatakse hetkel parimat objektide tuvastamise konvolutsioonilist närvivõrku YOLOv3 (ingl k. You Only Look Once). Selle töö väliselt on simulatsioonitarkvaradega AirSim ning Unreal Engine loodud lumine metsamaastik koos inimestega erinevates kehapoosides. YOLOv3 närvivõrgu treenimiseks võeti simulatsioonist välja vajalikud andmed, kasutades skripte. Lisaks muudeti närvivõrku, et lisaks inimese asukohta tuvastavale piirikastile väljastataks ka inimese kauguse ennustus. Antud töö tulemuseks on mudel, mille ruutkesmine viga RMSE (ingl k. Root Mean Square Error) on 2.99m objektidele kuni 50m kaugusel, säilitades samaaegselt originaalse närvivõrgu inimeste tuvastamise täpsuse. Võrreldavate meetodite RMSE veaks leiti 4.26m (teist andmestikku kasutades) ja 4.79m (selles töös kasutatud andmestikul), mis vastavalt kasutavad kahte eraldiseisvat närvivõrku ning LASSO meetodit. See näitab suurt parenemist võrreldes teiste meetoditega. Edasisteks eesmärkideks on meetodi treenimine ning testimine päris maailmast kogutud andmetega, et näha, kas see üldistub ka sellistele keskkondadele.

Võtmesõnad:
kaamera, objektide tuvastamine, sügavusinfo ennustamine, kovolutsioonilised tehisnärvivõrgud, YOLOv3, simulatsioon.

CERCS: T111 Pilditehnika
P176 Tehisintellekt

Acronyms and Abbreviations

Abbreviation	Full form
LiDAR	Light Detection and Ranging
NN	Neural Network
CNN	Convolutional Neural Network
YOLO	You Only Look Once
SSD	Single Shot Detector
bbox	Bounding Box
RMSE	Root Mean Square Error
mAP	Mean Average Precision
COCO	Common Objects in Context
IoU	Intersection over Union

1. Introduction

The complexity of tasks performed by robots, and the degree of their autonomy and self-learning capabilities, have been steadily increasing. There is an immediate and pressing need to ensure that the increasing deployment of robot workers does not bring new safety risks for human workers [2]. Machines have taken over the boring monotonous tasks that were previously performed by humans. But humans cannot be completely removed from the process, at least not yet. When a human is driving, the decisions are based on visual information that is received using eyes, and then decision is made to whether accelerate, turn or break, how fast to break, etc. Autonomous cars also need a way to be able to see what is around them, to perceive their surrounding. Object detection, mostly done using machine learning is used to help autonomous vehicles perceive their surrounding like humans. Autonomous vehicles are believed to be the answer for reducing pedestrian fatalities as most of such accidents are caused due to human negligence [3].

When talking about autonomous driving in urban environment we expect there to be roads, side-walks, traffic signs etc. These are all visual clues that can be and are read to aid in autonomous driving. This problem becomes much more complex when moving from urban to off-road environment. The visual clues are different, and harder to differentiate. Or as a contrast, when we are in middle of desert there are no clues. Machine learning techniques need to be trained with examples in order for them to learn, and when provided with a scenario they can predict based on examples they were trained on.

1.2. Motivation

Human safety is a crucial part of autonomous driving, and in order to avoid any injuries, detecting humans around the vehicle is one of the important aspects of perception in autonomous driving. As the computational resources onboard the vehicle are limited, we need to make this process not just reliable but fast and not so computationally expensive either. Demand for autonomous vehicles in hazardous environments is also increasing, like fire-fighting, protest monitoring and control, battlefields, etc. In such environments we have to make sure that no harm comes to nearby humans and that they are perfectly safe when around autonomous vehicles. While machine learning models have been developed for urban autonomous driving, there is little to no work done in training machine learning models for these scenarios. In addition, there are no datasets available with humans in off-road environments to train models. Also commonly used range sensors like LiDAR and stereo have their issues of sparse data and short range detections, respectively. This work takes a step towards solving the problem of human detection and distance estimation using an artificial neural network.

1.3. Objectives

Main objective of this work is to detect humans in a forest environment, estimate their distance using a monocular camera. This problem is addressed by using a simulation dataset where humans, with poses ranging from running to lying down, are in a snowy forest environment. The task is to modify an already existing machine learning algorithm, YOLOv3 to estimate the distances of all the detected humans while maintaining the current level of detection performance.

1.4. Organization of Thesis

- **In current chapter:** Introduction, motivation, and objectives of this work.
- **Background:** Literature review of similar work will be presented, overviewing traditional and recent state of the art approaches for this problem. Also core concepts about object detection and artificial neural networks.
- **Dataset used:** This chapter will include list of suitable real-datasets available, their shortcomings and limitations and, simulation data that was used.
- **Work done:** Detailed discussion of work done during this study, preprocessing of data and modifications made in YOLOv3.
- **Results and comparison:** In this chapter results of this work are presented. Comparison between two networks that were trained on using same configuration but on two different datasets.
- **Future work:** In this chapter some changes are suggested that can be made for better training and generalization of model.
- **Conclusion:** In last chapter conclusion of the work with short summary of lessons learned is presented.

2. Background

2.1. Milrem robotics and off-road autonomous driving

Milrem Robotics (MR) is an Estonian defence and civilian industry company with the primary focus of manufacturing unmanned ground vehicles (UGVs)[4]. THeMIS is an UGV developed by MR that weighs 1450 kg with maximum payload capacity of 750 kg at top speed of 20 km/h [5] and its main use case if off-road driving. The autonomy department of MR is working towards bringing off-road autonomous driving to a reality and aid in hazardous environments like firefighting, marshes, mapping unsafe mines, etc. One of the main focus of this work is human safety and for that detecting humans is an important aspect. This thesis is inspired to work towards a viable solution of detecting humans near the UGV and estimating their distances using only monocular camera. THeMIS can be remote controlled or autonomous, meaning there can be humans in vicinity that the machine has to be aware of and guarantee their safety at all times.

2.2. Object Detection

Object detection is computer technology that makes use of computer vision and image processing to detect instances of and object in a digital image or a video [6]. With a growing interest in automating the industry and autonomous vehicles, it has become a major area of exploration in computer vision. In this thesis object detection has been divided into two types, classical methods and state-of-the-art approaches that show much better results than the classical ones.

2.2.1. Classical Methods

Classical detection methods can be traced back to 1960s, the earliest application being pattern recognition systems used for character recognition [7]. They model interaction between locations of objects, and their context using manual engineering by defining structural relationship between objects [8].
Some of the classical methods that are widely used include Haar-like features [9], SIFT [9], SURF [9], GLOH [9], [10] and HOG detectors [10].

2.2.2. State-of-the-art methods

Artificial neural networks (explained in section 2.4), are state-of-the-art in object detection and that is why are widely used. In 2012, Neural Networks caught attention of researchers after ImageNet 2012 results. ImageNet can be called as “Annual Olympics of Computer Vision” [11]. ImageNet is a dataset consisting of 3.2 million annotated images and more than 5000 classes and was released in 2009 and a competition was launched with same name in 2010 with sole purpose of predicting if an object is present in an image or not (classification)

[12]. In 2012 a team from University of Toronto won with huge margin when compared to runners up. Their detection error was only 16.4% while the second in place had 26% detection error [13]. The winning team used deep convolutional network to perform the detection [13]. Neural networks have been improving ever since and in 2015 for the first time classification accuracy of humans was overtaken by a machine [11] (figure 2.1).

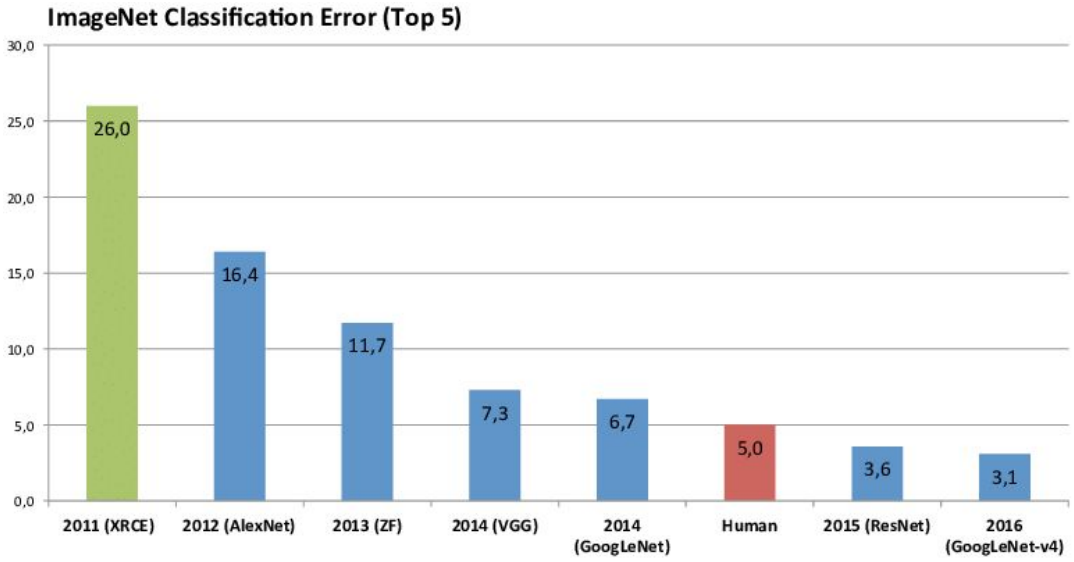

Figure 2.1. *Results of ImageNet competitions from 2010 to 2016* [11]

Some of the commonly used neural networks for the task of object detection are Faster R-CNN [11], SSD [11], RetinaNet [14] and YOLOv3 [11]. The widespread use of these networks is due to their well proven performance with object detection tasks. Figure 2.2 provides a time to precision comparison of them.

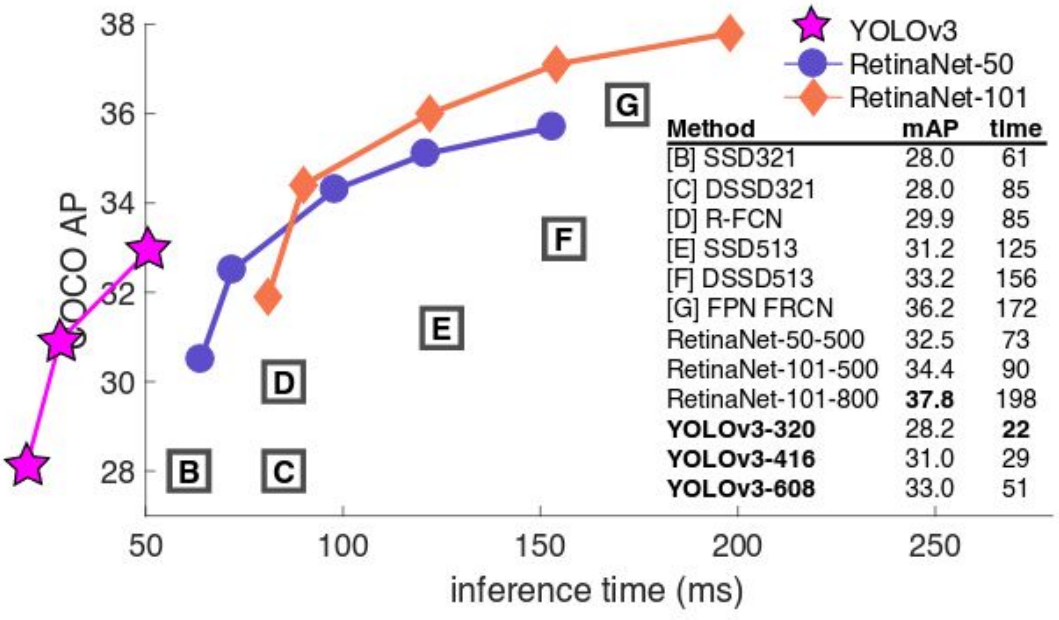

Figure 2.2. *Comparison of different object detection networks [11]*

2.2.4 Detection metric

Object bounding box predictions into a certain class can be divided into four categories as shown in table 2.1. If the prediction human was actually a human, it is TP prediction, if prediction human actually was not a human, it is FP prediction. If not human prediction actually was a human, it is FN prediction and lastly if not human prediction was not a human, it is TN prediction [15]. Based on this, other metrics can be formed.

Table-2.1. *Prediction matrix*

	Prediction: Human	**Prediction: No Human**
Actual: Human	true positive (TP)	false negative (FN)
Actual: No Human	false positive (FP)	true negative (TN)

Figure 2.3 shows data distribution of all the predictions. Precision is an evaluation metric that helps in understanding what proportion of predictions were correct. Similarly recall tells us what proportion of actual objects were identified correctly.

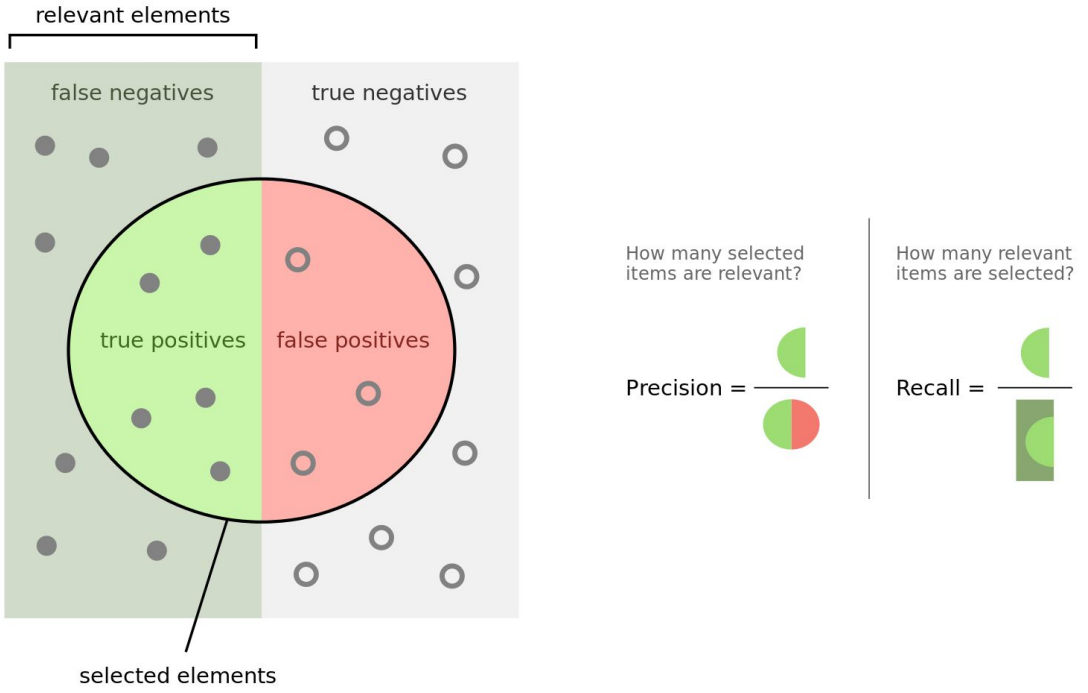

Figure 2.3. *Precision vs Recall [16]*

Mean average precision (mAP) has different definitions. One, used in this work, calculates average precisions (APs) over 11 recall values from 0-1.0 with increment of 0.1 (equation (1)) and is calculated over entire dataset. Where AP is summed over all the classes. It uses intersection over union (IoU) which is overlap between two bounding boxes for a set

threshold (figure 2.4). If IoU value is greater than set threshold (0.5 used) then detection is considered correct or true positive.

$$AP = \frac{1}{11} \times (AP_r(0) + AP_r(0.1) + ... + AP_r(1.0)) \quad (1)$$

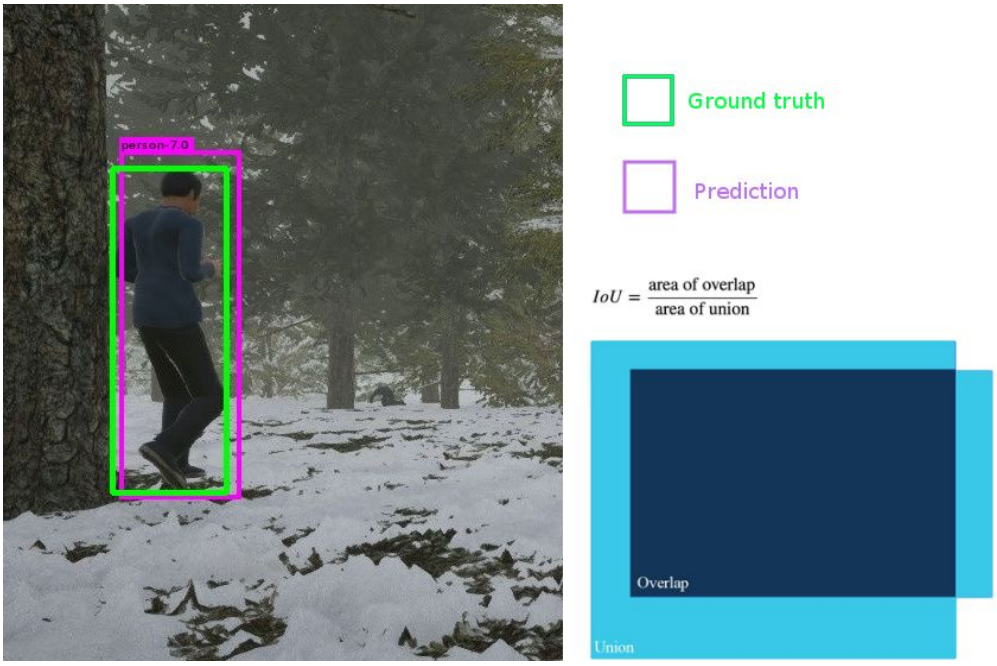

Figure-2.4. *IoU definition*

2.3. Distance Estimation

In many computer vision applications precise depth values are of crucial importance [17]. In recent years this task has gained attention due to industrial demand [17]. Other computer vision tasks, such as 3D object detection and tracking, 2D or 3D semantic segmentation and SLAM (Simultaneous localization and mapping) can exploit these accurate depth cues, that would result in better accuracy in mentioned tasks [17].
Next, brief introduction of currently often used sensors or methods that produce distance estimations is done.

2.3.1. LiDAR

LiDARs (Light detection and ranging) were first used on cars in 2000s and was made famous by Stanley [18]. LiDAR is a sensor based on time-of-flight principle and the idea was first introduced in 1930s [19]. Figure 2.5 explains working of a LiDAR, and it goes as follows [20].

1. A signal pulse (laser) is emitted from the sensor and precise time is recorded.
2. A reflection of that pulse is detected and again precise time is recorded.

3. Using the constant speed of light, the delay can be converted into distance (slant range) between source (sensor) and detected object.
4. Knowing the position and orientation of the sensor, the XYZ coordinates of the reflective surface can be calculated.

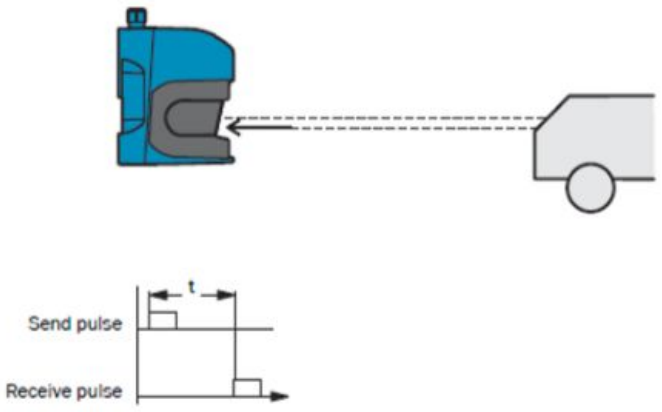

Figure 2.5. *Working principle of LiDAR [21]*

The output of LiDAR is a sequence of points consisting of three values: distance to surface, scanning angle and reflectivity [21]. The reflectivity of a surface depends on its structure, material and color. The most commonly used representation of LiDAR data is point cloud due to wide range of open source libraries. Figure 2.6 shows point cloud visualization.

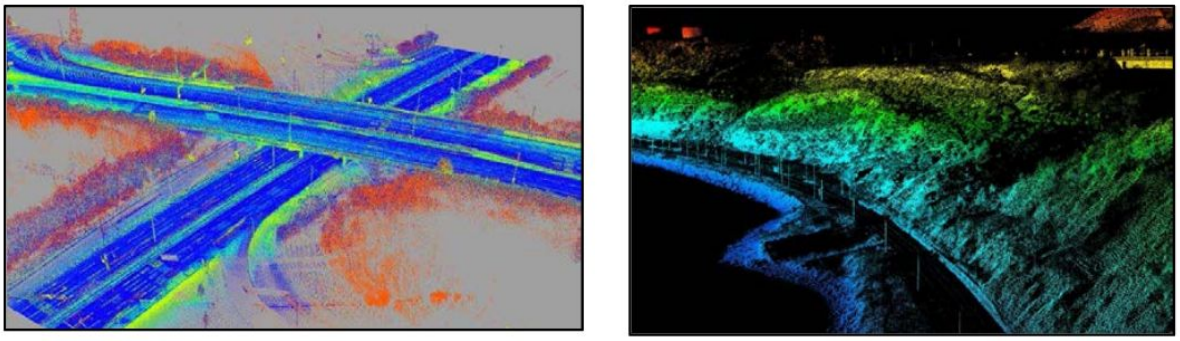

Figure 2.6. *Lidar data (point-cloud) collected from vehicle (left), and from a boat (right) [20]*

3D spinning LiDARs are widely used in autonomous driving these days. They commonly have 16, 32, or 64 beams with vertical field-of-view (FoV) of up to 30 degrees and horizontal FoV of 360 degrees [22]. The LiDAR generates a point cloud of its surrounding, but limited amount of scan lines results in high sparsity [17].

2.3.2. Stereo Camera

The human binocular vision perceives depth by using stereo disparity which refers to the difference in image location of an object seen by the left and right eyes, resulting from the eyes' horizontal separation [23]. The brain uses this binocular disparity to extract depth information from the two-dimensional retinal images which are known as stereopsis [23].
A stereo camera is a type of camera with two or more image sensors. Figure 2.7 shows camera geometry for binocular stereo system. Two cameras with parallel optical axes are arranged in a straight line commonly known as *baseline [24]*. Where F_L and F_R are the focal lengths of left and right cameras, respectively. And P is the point of which depth is being measured. Line F_L-P is seen as a point in left image but it appears as a line in right image and is called epipolar line [24].

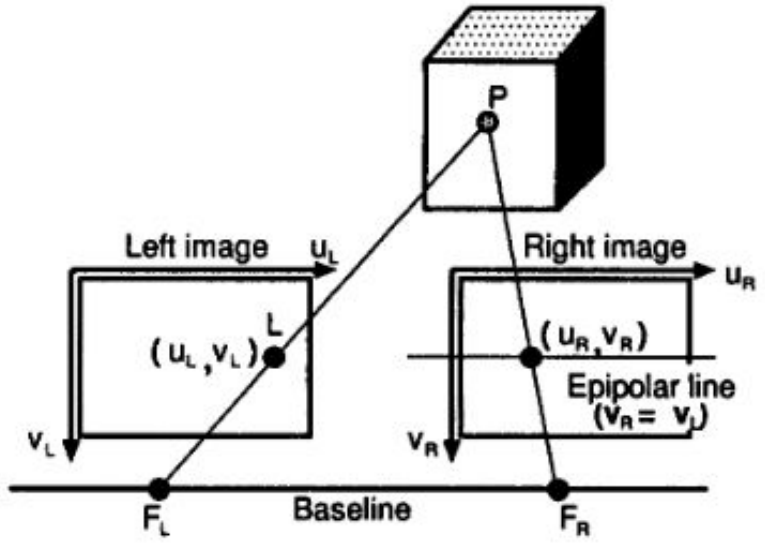

Figure 2.7. *Camera geometry for binocular stereo system[24]*

A stereo correspondence algorithm matches pixels from two images and returns displacement as disparity, which in proportional to the depth of that pixel [25]. Hence retrieving the third dimension that is important in many robotics applications. Stereo correspondence algorithms can be grouped into those producing sparse and those giving dense output [25]. Feature based methods like growing seed [26] are inspired from human vision studies and consist of matching segments or edges between two images, and they result in a sparse output, while stereo correspondence, matching pixels from two images produces dense output [25]. Robotic applications demand more and more dense output (figure 2.8), where dense map contains depth values for almost all real-world coordinates that are in scope of camera [27]. For outdoor application, distance estimation for longer ranges is required, and error rate of stereo-cameras increase with increase of distance [25].

Figure 2.8. *Top: RGB scene. Middle: sparse depth image. Bottom: dense depth image* [28]

2.3.3. Others

Among other sensors, time-of-flight (ToF) camera can also provide distance information. A 3D time-of-flight camera works by illuminating the scene using a modulated light source and then observing the reflected light [29]. ToF cameras produce a depth image, where each pixel is encoded with distance of corresponding point of the captured scene [29]. The distance range is limited by power of light beam shot and it is also susceptible to motion blur [29].

There have also been recent attempts to form a depth image from a monocular image, such models tried to predict depth directly from image. [30] introduced structure in learning framework, the respective motion of objects and robot are modeled as independent transformations - translations and rotations. But we cannot rely on distance values we get from a predicted depth image because they are not accurate enough to be usable in this application.

2.3.4. Evaluation metric

Root mean square error (RMSE) equation (2) tells us how spread out our data is from our mean, where $distance_{actual}$ is ground truth and $distance_{prediction}$ is the predicted distance, and N is total number of detections. It is commonly used in regression analysis to verify experimental

results. Since distance estimation is a regression problem, RMSE can be used for evaluation [31].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (distance_{actual} - distance_{prediction})^2}{N}} \quad (2)$$

2.4. Artificial neural networks

Artificial neural networks are an attempt in simulating the network of neurons similar to human brain so that computers can learn and make decisions in a similar manner as humans [32]. Following are some concepts for better understanding of neural networks and work done in this thesis.

2.4.1. Machine Learning

Machine learning (ML) is the study of algorithms and statistical models that can be used by computers to effectively perform tasks without programmed instructions, relying on examples instead [33].

ML algorithms learn from data. The data needed for learning should contain features and ground truths or detection results about those features. In image data, features are the pixels of an image. In some cases these features need to be engineered, extracted or modified to better suit the needs of training. Training is the process of an ML algorithm, where examples (dataset) are used to teach the model. Dataset is often split into training, validation and test sets. Training set is the data that is directly fed to ML algorithm as examples. Validation set is used to indicate during training when overfitting might start and also helps in choosing what hyperparameters or models produced the most generalizable result. Test data is used to generate the final results of proposed algorithm and to measure the accuracy. Overfitting and underfitting are phenomenons where either model is unable to generalize on new data or is unable to generalize on training set [34].After a model is trained a scoring function is applied to evaluate the performance of trained model [35]. This trained model can then perform predictions on new unseen data (figure 2.9).

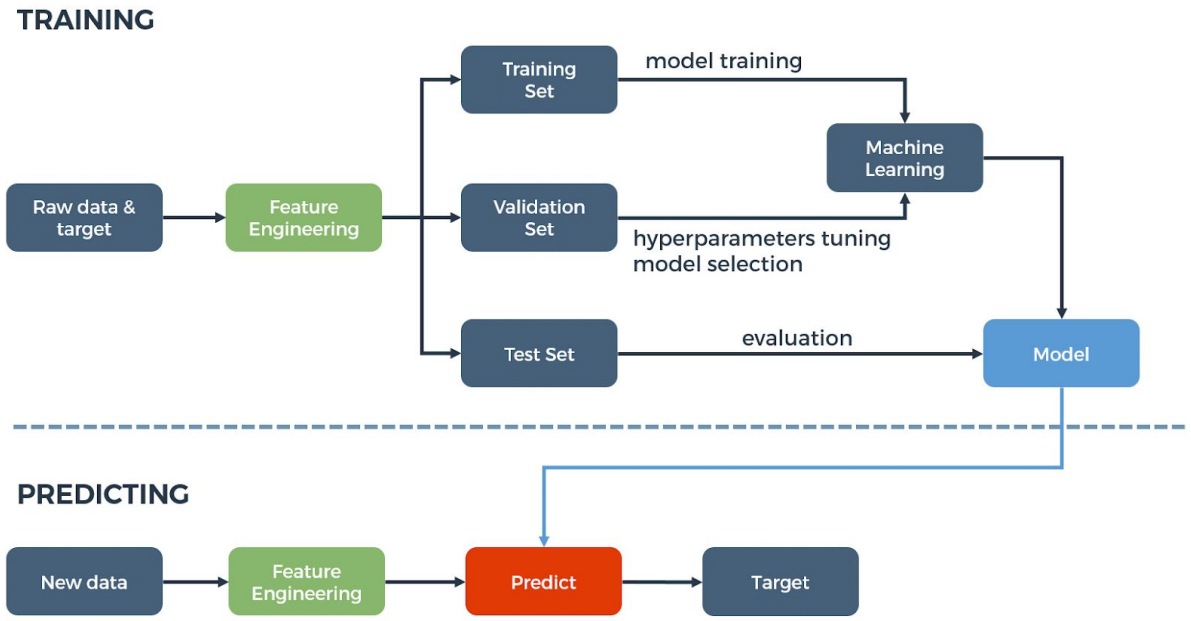

Figure 2.9. *Machine learning process [36]*

2.4.2 Neural network structure

Basic building block of a neural network is a neuron (figure 2.10). Each neuron takes a vector input x_i, multiplies it with a weight w_i and adds a bias b to it. After that a non-linear activation function f is applied to give an output from that neuron [37]. In terms of nervous system, activation function decides whether a neuron will be fired or not, or whether it will be activated. Activation function can get any value from +infinity to -infinity, and performs certain mathematical operation on it [37]. Deep neural networks are multi-layered structures (figure 2.11) where each layers has a fixed number of neurons. The number of neurons in the first layer is equal to size of input vector [37], and that of output layer equals the number of classes.

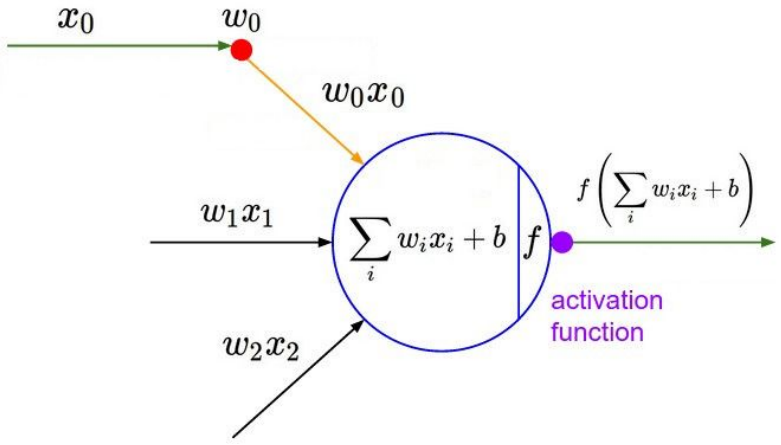

Figure 2.10. *architecture of a single neuron [37], where x is input, w is weight, b is bias, f is activation function*

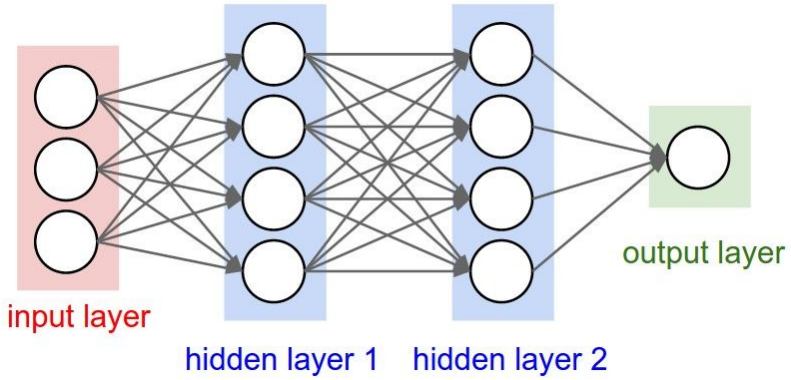

Figure 2.11. *Three layered neural network with two fully connected layers[39]*

For object detection, image pixels are used as features while training a network or making a prediction. If we use fully connected layers, where every neuron is connected to all the neurons in the previous layer then e.g. for a single image of size 32x32x3, a single neuron in the first fully connected layer will have 3072 weights. The number of weights will grow to huge numbers just because we have that many pixels in an image [37]. This calls for a better solution that can bypass those huge computations but still give us desired results.

Convolutional Neural Networks (CNN) are type of neural network that also comprise of neurons, have weights and biases, activation functions, etc. Convolutional neural networks consist of one or more convolutional layers, which apply convolution functions from computer vision [38]. Convolutional layers learn to extract useful features (curves, edges, shapes, textures) from images, and are location invariant that makes them favored option for image detection tasks [37]. Convolutional layer uses convolutional filters of size *axa* that are moved along the image by fixed size (stride) to output a feature map containing features of image.

2.4.3 Neural network training

During training process loss refers to the error of the model and shows how well model is improving. If loss is decreasing model training is going well, if it becoming constant means model is already trained for set parameters [34]. An activation function (Softmax, ReLU, sigmoid, etc) is applied to output of neurons to make them non-linear. There exist different optimizer functions (SGD, AdaGrad, Adam, etc) that tune the weights of network using gradient descent resulting in reduction of loss and better predictions [39]. Data is fed to algorithm in form of batches where a batch can have multiple training examples. Training over a single batch is known as one iteration cycle and going over the whole dataset once is called epoch [40].

2.5. YOLO (You Only Look Once)

Until 2015, the detection systems were repurposing the classifiers to perform detection, meaning they used to look at specific regions of image and make predictions based on every region, in a sliding window manner [16]. YOLO formulated object detection as a regression problem to spatially separate bounding boxes and associated class probabilities. YOLO is a single neural network that predicts bounding boxes and predicted class probabilities directly from full image, and in one evaluation [41]. Since this whole detection pipeline consists of a single network, it can be optimized end-to-end directly on detection performance [41]. There are three YOLO versions, YOLO [41], YOLOv2[42], [43], and YOLOv3 [1].

2.5.1. YOLO

YOLO does not use sliding window approach, searching over regions of image. Instead it sees entire image during training and testing and encodes contextual information about classes and their appearance. Fast R-CNN [44], a top detection method of that time has better accuracy when compared to YOLO but it is considerably slower and has twice the background error when compared to YOLO[41] .

YOLO is a CNN and has 24 convolutional layers and two fully connected layers. Initial convolutional layers extract features from image and fully connected layers predict class probabilities and bounding box coordinates [41].

2.5.2. YOLOv2:

Moving from YOLO to YOLOv2 there are some major changes (table 2.2) introduced to network structure to improve detections[42], [43]. Following are few design changes that make YOLOv2 better than YOLO.

1. State-of-the-art detection algorithms make use of classifier pre-trained on ImageNet, and most classifiers operate on smaller input images of size *256x256*. YOLOv2 is first fine-tuned with classification network at *448x448* resolution for 10 epochs over whole ImageNet dataset [43] making it higher resolution classifier. This gives time to adjust filters to work better with higher resolution inputs. And the network is fine-tuned on detections, this gives an increase of almost 4% mAP when compared to YOLO [43].
2. YOLOv2 uses anchors on convolutional layer to make predictions, unlike YOLO that used fully connected layers to predict. Anchor is an initial estimate of size (width, height) of a bounding box. YOLOv2's convolutional layers downsample an image by a factor of 32 so by using input image of resolution 416 a feature map of *13x13* is generated. Switching to convolutional layer with anchors for prediction decreases mAP by 0.4, but decreases computation cost by 33% [43].
3. Anchors can either be provided randomly, or calculated from the training set using different clustering methods. The second option helps in speeding up the training. Otherwise the network will have to learn the actual anchors by its own. K-means

clustering has proven to be the most effective method when calculating anchors and gives the best average IoU [43]. Anchor boxes help in increasing recall.

4. YOLOv2 predicts detections on *13x13* feature map which is sufficient for larger objects, but we need fine feature maps if we need to detect smaller objects. YOLOv2 is solving that problem using a passthrough layer. The passthrough layer concatenates the higher resolution features with lower resolution features obtained from earlier convolutional layer of *26x26* resolution [43].
5. To make YOLO able to predict equally well on images of different resolutions the network changes every few iterations while training making training multi-scale. After every 10 batches network choses a new image dimension size. This forces the network to learn and predict well across variety of different resolutions [43].

Table 2.2. *Feature improvements from YOLO to YOLOv2, VOC2007 dataset [43]*

Feature	mAP increase from YOLO to YOLOv2
high resolution classifier	3.7
convolutional layer detection	-0.3
anchor boxes	0.4
passthrough layer	1.0
multi-scale	1.2

Original YOLOv2 implementation is based on Darknet [45], an open source neural network framework, written in C and CUDA [46]. YOLOv2 is also known as Darknet-19, and name comes from it having 19 convolutional layers.

2.5.3. YOLOv3

YOLOv3 adds some incremental changes to YOLOv2 to improve detection [1]. Following is a list of changes introduced in standard YOLOv3.

1. YOLOv3 predicts an objectness score for each bounding box using logistic regression and it gets score of 1 if it overlaps ground truth more than any other prediction.
2. Each bounding box predicts class it may contain using independent logistic classifier.
3. YOLOv3 predicts boxes at three different scales instead of just one (YOLOv2). Feature maps are upsampled and merged with feature maps from previous layers as well and new convolutional layers are added to process this feature map and predict. K-means is used for clustering to determine bounding box priors (anchors). 9 clusters are chosen and divided evenly across 3 scales, making 3 predictions each scale.
4. It uses Darknet-19, but the addition of residual connections and feature map merging make it larger, 53 convolutional layers, and hence it is called Darknet-53 (table 2.4). This makes network larger and predictions become slower when compared to YOLOv2, but it increases accuracy (table 2.3).

Table 2.3. *YOLOv2 vs. v3 [42], [43]*

Model type	mAP	FPS
YOLO v2	44.0	40
YOLO v3	58.0	20

Table 2.4. *Darknet-53 layers*

	Type	Filters	Size/Stride	Output
	Convolutional	32	3 x 3	256 x 256
	Convolutional	64	3 x 3/2	128 x 128
1 x	Convolutional Convolutional Residual	32 64	1 x 1 3 x 3	128 x 128
	Convolutional	128	3 x 3/2	64 x 64
2 x	Convolutional Convolutional Residual	64 128	1 x 1 3 x 3	64 x 64
	Convolutional	256	3 x 3/2	32 x 32
8 x	Convolutional Convolutional Residual	128 256	1 x 1 3 x 3	32 x 32
	Convolutional	512	3 x 3/2	16 x 16
8 x	Convolutional Convolutional Residual	256 512	1 x 1 3 x 3	16 x 16
	Convolutional	1024	3 x 3/2	8 x 8
4 x	Convolutional Convolutional Residual	512 1024	1 x 1 3 x 3	8 x 8
	Avgpool Connected Softmax		Global 1000	

2.6. Combined object detection and distance estimation

The problem of combined detecting of objects and estimating the distance is not researched much to this point, as explained in sec 2.1 and 2.2. One method that is using similar concept as one proposed in this work is DisNet [47]. It uses a neural network with 3 hidden layers in conjunction with YOLOv3 [47] to predict distance of a detected object from the camera [47]. A feature vector is made from output of YOLOv3 and is fed to DisNet to predict the distance of the detected object. Figure 2.12 explains working of DisNet with simple block diagram.

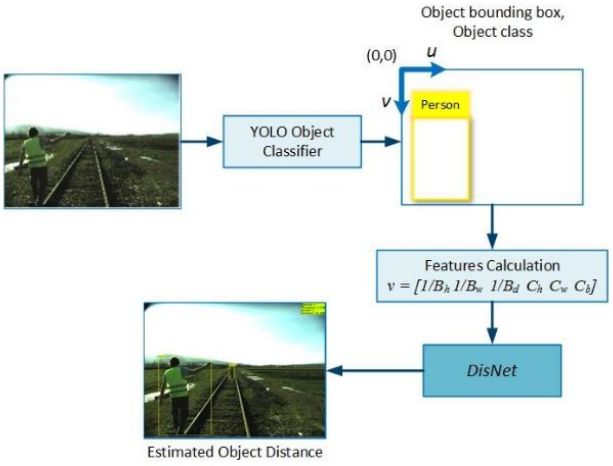

Figure 2.12. *DisNet based system for object distance estimation, where B_h, B_w, B_d being height, width and diagonal of bounding box, and C_h, C_w, C_d are average width, height and diagonal of class [47]*

3. Datasets

As explained in section 2.4, data is needed for training neural networks. Some of the available datasets, and the one used in this work are explained in the next sections.

3.1. Available Datasets

There are some comprehensive datasets available for object detection, but none for the specific task of also predicting the distance of detected object. Though some of these datasets provide range information (LiDAR data) as shown in table 3.1, but none of them focuses on or provides a good enough data that can be extracted for human detection, and secondly they cover urban driving scenarios, city environments, and sunny day. While main focus of this work is detection in off-road environment, snowy terrain, forest, which poses unique challenges. Hence the idea to make use of simulation data.

Table 3.1. *List of available datasets that are related to human detection [48]*

Dataset	Vision	Lidar	Large Scale	Seasonal Change	Off-road
New College Vision and Laser Dataset	Y	Y	Spatially	N	N
Rawseeds Project	Y	Y	No	N	N
CMU Visual Localization Dataset	Y	N	Spatially and Temporally	Y	N
Ford Campus Vision and Lidar Dataset	Y	Y	Spatially	N	N
Alderly Day/Night Dataset	Y	N	Spatially	N	N
Nordland Dataset	Y	N	Spatially	Y	Y
Malaga Urban Dataset	Y	Y	Spatially	N	N
VPRiCE Dataset	Y	N	Spatially	N	N
Kitti Dataset	Y	Y	Spatially and Temporally	N	N
Cross Season Dataset	Y	N	Spatially	N	N
MIT Stata Centre Dataset	Y	Y	Spatially and Temporally	Y	N
NCLT Vision and Lidar Dataset	Y	Y	Spatially and Temporally	Y	N

3.2. Dataset used

Simulation dataset used in this work was developed in-house during Nutikas project [49]. It was generated from a simulation environment that mimics snowy terrain and forest environment. AirSim [50] was used which gives us segmentation and depth images. AirSim is based on Unreal Engine [51] and it was populated by humans in different poses, having random trajectories. A car with camera was driven around to capture different scenes. Camera, segmentation and depth images we provided. Camera image is fed as an input to the network, segmentation image is needed to extract tight boundary (bounding box) around humans, and depth image is required to get distance of the human. Figure 3.1, 3.2, and 3.3 shows camera, segmentation and depth image at one instant. Images provided have *1280x720* resolution and whole dataset has 11 different poses (table-3.2) which makes detection problem much harder.

Figure 3.1. *Final scene image*

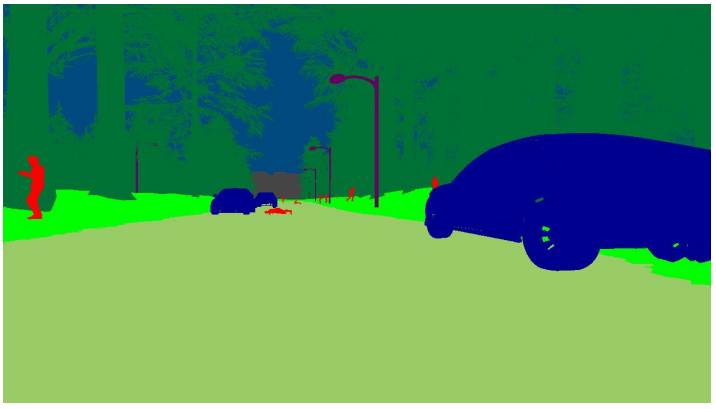

Figure 3.2. *Corresponding segmentation image for figure 3.1 where red color is for human class*

Figure 3.3. *Corresponding depth image for figure 3.1*

Table 3.2. *List of poses in simulation dataset*

No.	Human Pose
1	Standing
2	Running forward
3	Running backward
4	Walking
5	Crouching
6	Waving arms
7	Situps
8	Lying on ground
9	Misc. poses (3 poses)

4. Work done

Annotated data is required for training, so provided dataset was preprocessed keeping requirements in mind. YOLOv3 is modified to predict distance along with bounding boxes and error metric is added to evaluate the model.

4.1. Data preprocessing

The detection network implementation used, darknet [52] requires all the training images to be in a training folder and each of the image is required to have a corresponding text file with same name and *.txt* extension. Each row of text file refers to one bounding box and is arranged as follows.

Class_id bbox_centre_x bbox_centre_y bbox_width bbox_height

Where bbox refers to bounding box, bbox_centre_x and bbox_centre_y refer to x and y pixel coordinates of centre point of bounding box, bbox_width and bbox_height are width and height of bounding box in pixels (figure 4.1). All these pixel values are normalized with respect to image size. An extra column is added to carry the distance information, so format becomes

Class_id bbox_centre_x bbox_centre_y bbox_width bbox_height distance

Figure 4.1. *Bounding box annotation params*

Figure 4.2 shows the general architecture of data preprocessing. Since the class information comes from segmentation image, first thing to do is thresholding the segmentation image according to human class_id, to get human masks. If thresholding is skipped, we end up with incorrect blobs and smaller blobs (larger distances) are not detected, and also get blobs that are not human (trees, road, cars, etc). Thresholding is done for red color, since red is the human class. Once that is done, contour detection is applied to get the contours of humans in the masked image. This is followed by convex-hull [53] that helps to calculate border pixels for the red blobs or humans. From convex-hull values, it is easy to get the extreme pixel values (left, right, top, and bottom) that perfectly fit the human, those extremes are used to calculate the centre of bounding box, width and height, and is normalized using width and height of the input image.

Once bounding boxes are obtained, this information is then fused with depth image to get distance information of that bounding box. A cut-off of 30 and 50m distance was used. Human bounding boxes that had their distance values more than set threshold were discarded. All this information was then stored in an annotation file in *.txt* format. This process was repeated for all the images in dataset and takes ~2 seconds for one image.

Although this process for automating data saved a lot of time of manual labeling, it had corner cases (~20 %) in current dataset, that it did not cover. Figure 4.4 and Figure 4.5 show such examples where due to overlap between two humans and having white region surrounded by red (human class) can get confused as one contour.

Figure 4.3 has the visualization of bounding box drawn on masked image (the original image is depicted in Figure 3.1 and the segmented image in Figure 3.2).

A dataset consisting of 890 images, with text files is compiled.

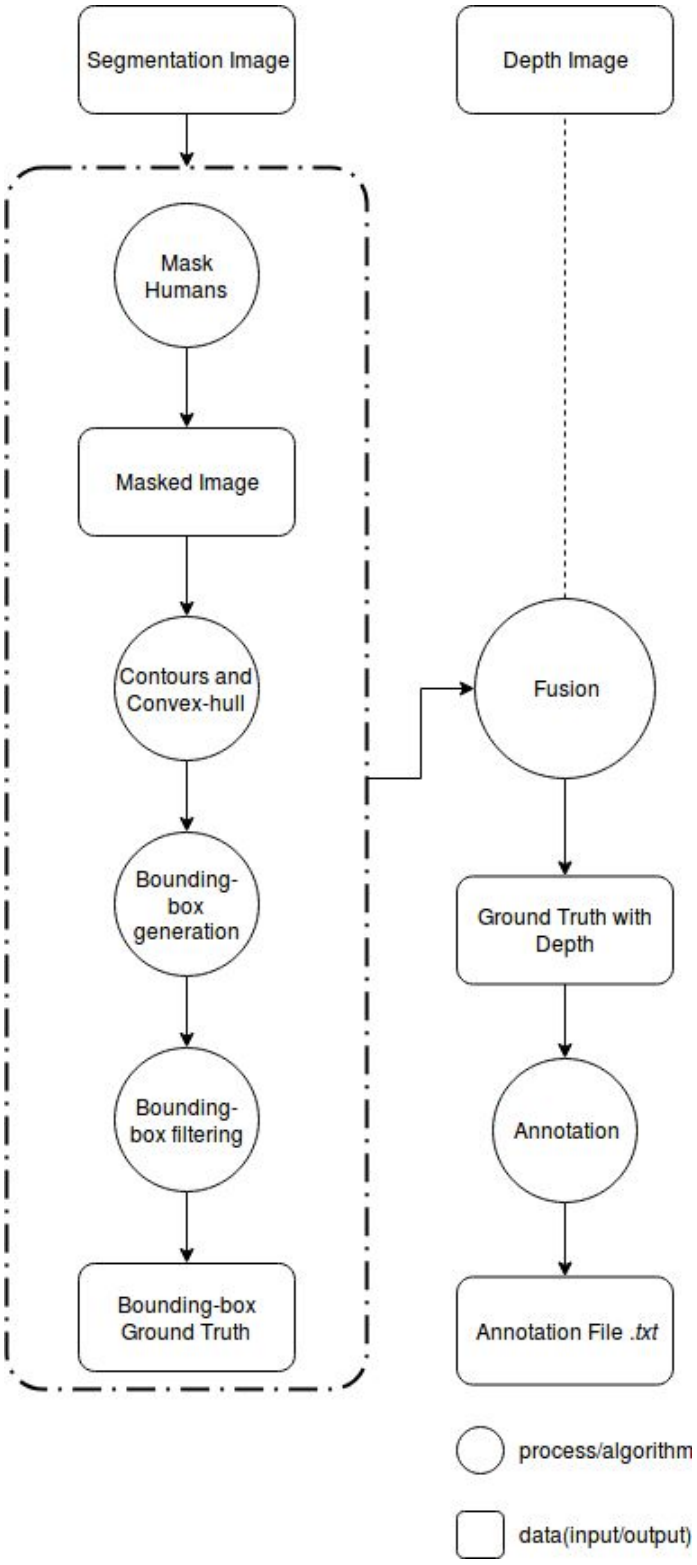

Figure 4.2. *The general architecture for annotating data in required format. Input images, and all the processes/algorithms are shown.*

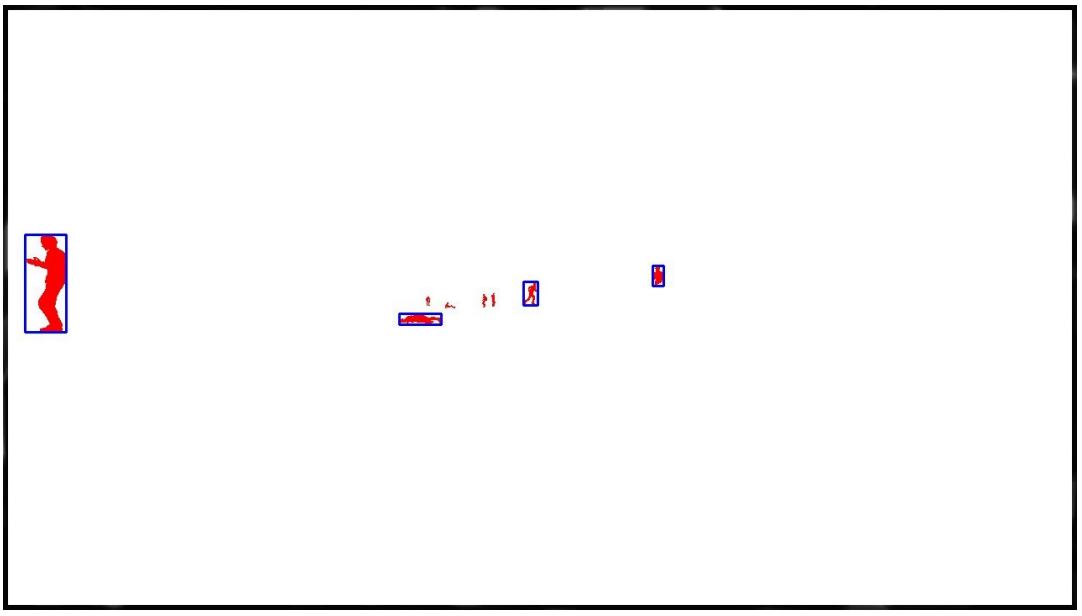

Figure 4.3. *The result of thresholding applied to figure 3.1, followed by contours and convex-hull and finding extremes. Distance cut-off was applied so any human farther than 50 meters is not made a bounding box of*

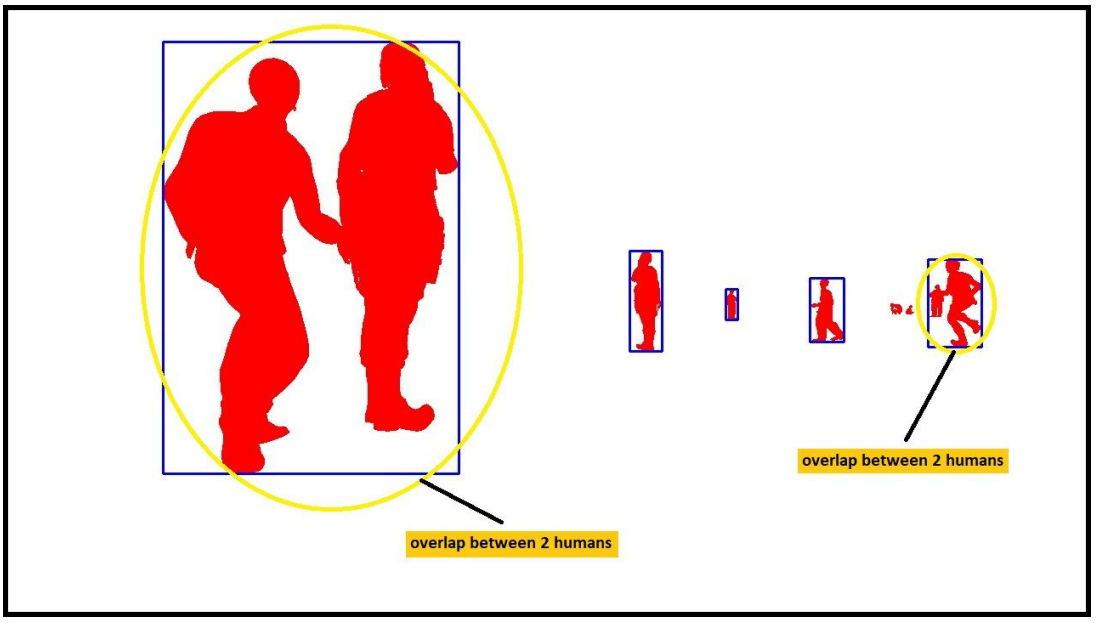

Figure 4.4. *When there is even slight overlap between two humans, they get same contour and hence we wind up with one bounding box instead of two*

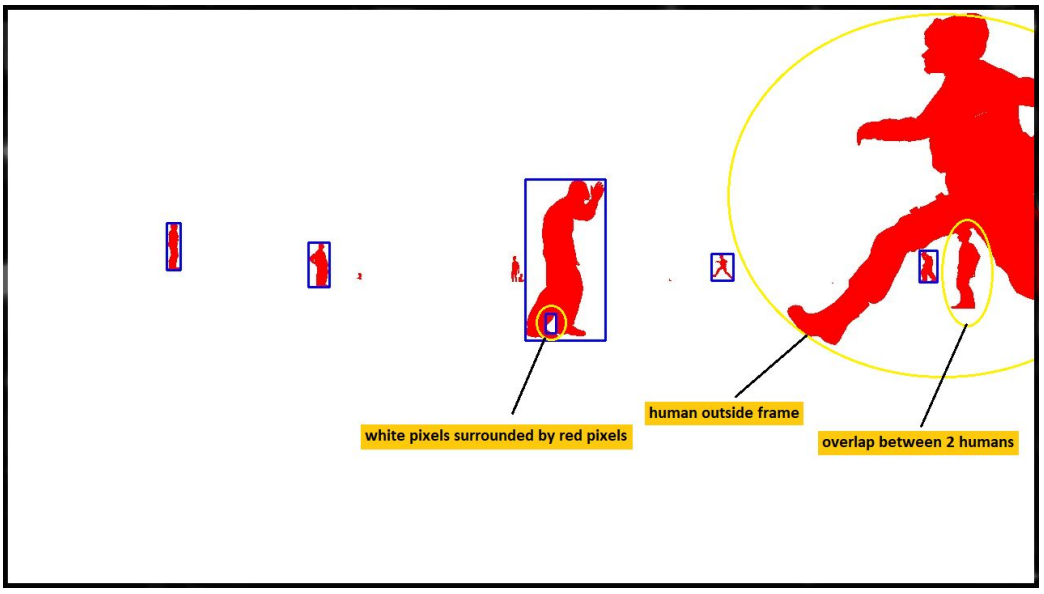

Figure 4.5. *shows that if a human is not fully in the frame then it does not get selected as a contour, and if there is significant gap between arms or legs it also get a bounding box*

4.1.1. Manual Labeling

Although automatic labeling pipeline is quick and saved us a lot of time, it did not cover a few cases (figures 4.4 and 4.5). Those remaining images (190) were manually labelled using YOLO-Annotation-Tool [54]. It is easy to use bbox labeling tool with user-friendly graphical user interface (figure 4.6). It also provides horizontal and vertical markers next to mouse cursor so it becomes easy to mark bounding boxes. It saves the pixel coordinates to a text file for each image. The text files with these pixel coordinates were subsequently converted into required training format using a custom script.

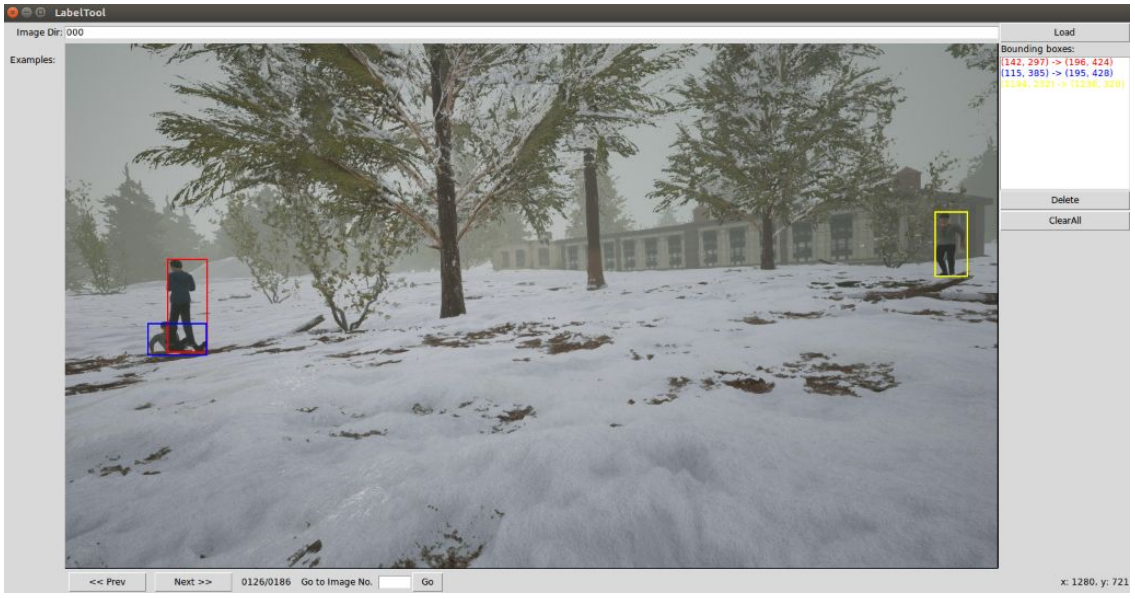

Figure 4.6. *Snippet of the tool used for manual labeling*

4.2. YOLOv3 modifications

As explained in section 2.4, YOLOv3 is designed to predict bounding boxes and certainty score of predicted object. In this work it is modified to accept an extra training input of distance and also predict it. The following functionalities were added to the YOLOv3 source code:

1. Reading distance information from the annotation file
2. Normalizing distance values after reading them from the annotation files. This yields lower loss and better results
3. Adding loss function to yolo layer to account for distance losses and adjust weights accordingly. YOLOv3 uses squared loss for each bounding box and class loss. Another function is added to compute loss for distance squared and added to previous loss (equation (3)).

$$Loss = bbox_{loss}^2 + class_{loss}^2 + distance_{loss}^2 \quad (3)$$

Where $bbox_{loss}$ comprises of all the individual losses for width, height, centre_x, and centre_y position of bounding box, as shown in equations (4)-(7)

$$centre\_x_{loss} = scale * (centre\_x_{actual} - centre\_x_{predicted}) \quad (4)$$
$$centre\_y_{loss} = scale * (centre\_y_{actual} - centre\_y_{predicted}) \quad (5)$$
$$width_{loss} = scale * (width_{actual} - width_{predicted}) \quad (6)$$
$$height_{loss} = scale * (height_{actual} - height_{predicted}) \quad (7)$$

Where *scale* is a constant that can help with how aggressively model weights are changed with loss values. Similarly losses for class and distance are shown in equations (8) and (9), respectively.

$$class_{loss} = class\_probability_{actual} - class\_probability_{predicted} \quad (8)$$
$$distance_{loss} = scale * (distance_{actual} - distance_{predicted}) \quad (9)$$

Where actual class probability is 1.0 and predicted class probability is between 1.0 and 0.0. This scale parameter in equation (9) was tweaked for different values ranging from 1-18 to get better RMSE values for distance evaluation.

4. Modification of yolo layer (detection layer) to predict distance, and to account for distance loss in order to change the weights
5. More weight was given to distance loss function to achieve better RMSE for distance with minor trade-off (~1%) for bounding box mAP
6. RMSE function was added to get overall distance accuracy of model
7. Some other minor modifications to help with evaluation with dataset used

5. Results

5.1. Training Procedure

Models were trained using a varying learning rate to get better results quicker. Table 5.1 shows best combination for given dataset found by observing the loss graph.

Table 5.1. *Learning rates used*

Learning rate	Number of iterations
0.001	0- 2000
0.0001	2000-5000
0.00001	5000-10000
0.000001	10000-18000

Two datasets were extracted from provided data by limiting the maximum distance of humans to be annotated to 30m (dataset-30) and 50m (dataset-50). 525 training images were used, with 175 validation images and 190 test images. Two models were trained using same parameters but different datasets. For training on both of these datasets, distance loss was multiplied by 12.0, as it showed best results.

Data augmentation was used to increase amount of training data without incurring any extra labeling cost. Standard data augmentation techniques like changing saturation (1.5), exposure (1.5), hue (0.1), cropping, flipping of images were used. These were already provided in darknet implementation. Figure 5.2 and 5.3 show loss variation for training on dataset-30 and dataset-50 throughout the training process. The sharp curve in the beginning is due to higher learning rate (table 5.1), as loss varies more with higher learning rates. Both trainings were terminated before set iterations because loss had become constant. Leaky RELU (rectified linear unit) (figure 5.1) was used as activation function, and AdaGrad, a sophisticated gradient descent algorithm that can rescale gradients of each parameter [34] was the optimizer used.

$$f(x) = \begin{cases} x & if \;\; x > 0 \\ 0.01x & otherwise \end{cases}$$

Figure 5.1. *Leaky RELU activation function [55]*

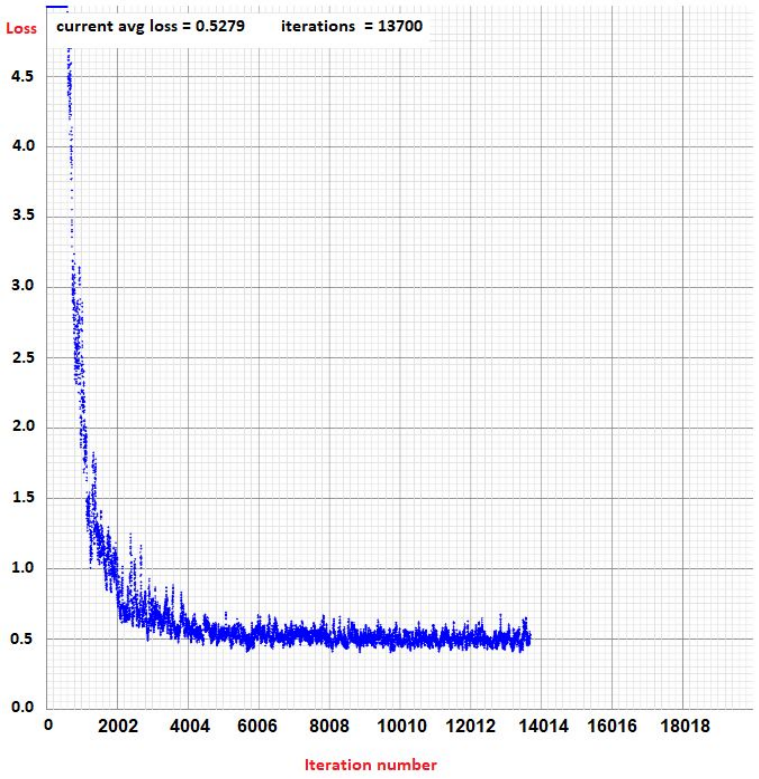

Figure 5.2. *Loss variation for training on dataset-30*

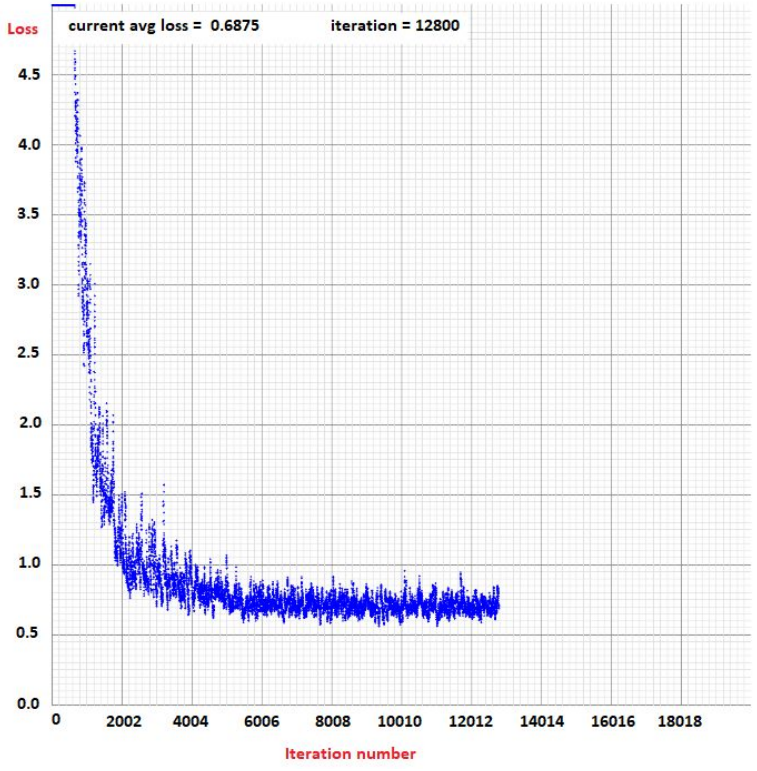

Figure 5.3. *Loss variation for training on dataset-50*

A higher learning rate is used in the beginning to decrease loss and make model converge quickly, and we shift to lower learning rates after that so we do not overshoot the minima. If we use smaller learning rate from beginning model will take a lot of time to converge to the minima.

5.2. Evaluation Metrics

RMSE is used to get distance accuracy of the model and COCO evaluation metric [56] is used for bounding box accuracy, explained in table 5.2. Evaluation results are presented for two models as mentioned previously.

Table 5.2. *COCO evaluation metric explanation [56]*

Average Precision (AP):	
AP	% AP at IoU=.50:.05:.95 **(primary challenge metric)**
$AP^{IoU=.50}$	% AP at IoU=.50 (PASCAL VOC metric)
$AP^{IoU=.75}$	% AP at IoU=.75 (strict metric)
AP Across Scales:	
AP^{small}	% AP for small objects: area < 32^2
AP^{medium}	% AP for medium objects: 32^2 < area < 96^2
AP^{large}	% AP for large objects: area > 96^2
Average Recall (AR):	
$AR^{max=1}$	% AR given 1 detection per image
$AR^{max=10}$	% AR given 10 detections per image
$AR^{max=100}$	% AR given 100 detections per image
AR Across Scales:	
AR^{small}	% AR for small objects: area < 32^2
AR^{medium}	% AR for medium objects: 32^2 < area < 96^2
AR^{large}	% AR for large objects: area > 96^2

Human distribution in dataset with respect to area as defined by COCO metric is shown (table 5.3).

Table 5.3. *Area distribution of dataset-30 and dataset-50*

Dataset	Area distribution
dataset-30	Small: 0.09 Medium: 0.715 Large: 0.19
dataset-50	Small: 0.37 Medium: 0.5 Large: 0.125

As expected we have more smaller humans in 50m cut-off than in 30m. Table 5.4 shows the RMSE values of distance for both models for different distances. It shows that model trained

on dataset-30 gives quite good results for larger distances as well, while doing pretty good for distances under 10m.

Table 5.4. *RMSE values for both models*

Dataset	RMSE (0-10m)	RMSE (0-20m)	RMSE (0-30m)	RMSE (0-40m)	RMSE (0-50m)
dataset-30	0.89m	1.71m	1.97m	2.98m	3.21m
dataset-50	1.4m	1.37m	2.24m	2.63m	2.99m

Table 5.5 shows results after evaluating model trained on dataset-30 and dataset-50 on COCO metric. The main metric is AP at 0.5 IoU, that shows model performance on detections. There is not much drastic difference in these two trainings except AR for small areas that comes from dataset-50 having many small examples to train on (table-5.3).

Table 5.5. *Detection results*

Metric	IoU	Area	maxDets	dataset-30 values	dataset-50 values
Average Precision (AP)	0.50:0.95	all	100	0.512	0.466
Average Precision (AP)	0.5	all	100	0.878	0.856
Average Precision (AP)	0.75	all	100	0.581	0.443
Average Precision (AP)	0.5:0.95	small	100	0.241	0.368
Average Precision (AP)	0.5:0.95	medium	100	0.544	0.509
Average Precision (AP)	0.5:0.95	large	100	0.547	0.571
Average Recall (AR)	0.5:0.95	all	1	0.343	0.221
Average Recall (AR)	0.5:0.95	all	10	0.604	0.573
Average Recall (AR)	0.5:0.95	all	100	0.604	0.575
Average Recall (AR)	0.5:0.95	small	100	0.378	0.498
Average Recall (AR)	0.5:0.95	medium	100	0.621	0.614
Average Recall (AR)	0.5:0.95	large	100	0.650	0.645

Prediction result (figure 5.4) and ground truth (figure 5.5) are shown for one of the test images in form *class-distance*. Model was also tested on real world data (figure 5.6).

Figure 5.4. *Prediction results*

Figure 5.5. *Ground truth for figure 5.4*

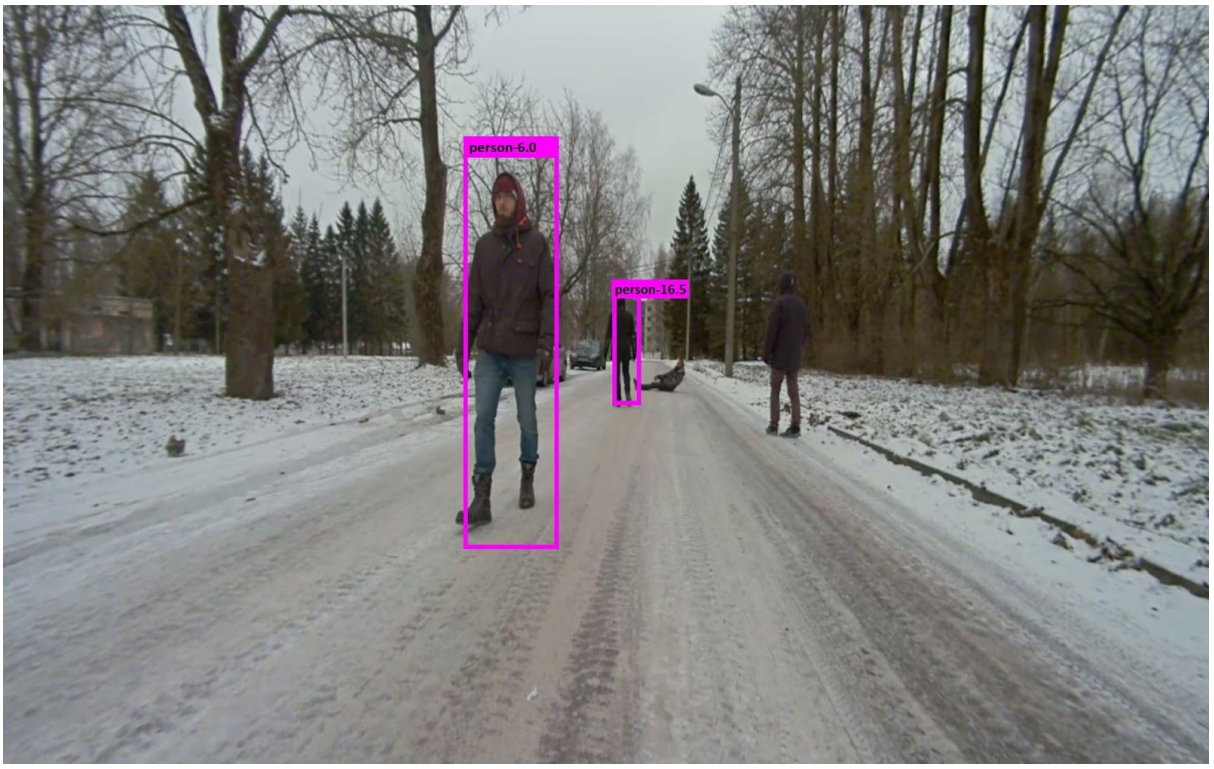

Figure 5.6. *Real-world image tested on trained model*

5.3. Comparison

Proposed technique and DisNet [53] both rely on YOLOv3 [53] for detection and are estimating distance differently. DisNet does not provide any metric to gauge performance of distance estimator, but from result table a distance of 50m showed the result of 54.26m, showing an error of 4.26m [53] while we have RMSE of 2.99. Also least absolute shrinkage and selection operator (LASSO) model made by Nutikas [49] uses a feature vector of height, width, and area (in pixels) of a bounding box predicted by YOLOv3 and predicts its distance. It was trained on dataset-30 and dataset-50 and then tested on same test set. It showed more than 60% RMSE increase (table 5.7) than proposed model.

Table 5.7. *Comparison of distance RMSE with LASSO model*

Dataset	RMSE (m)	% RMSE increase
dataset-30	3.28	66.5
dataset-50	4.79	60.2

6. Future Work

YOLOv3 was trained using only simulation data which was limited in a sense that most humans were in similar clothing, physique, skin tone and were all adults. In order to get better understanding of behaviour more variance is needed. Even with these limitations, it provides proof-of-concept that this approach is viable and could be made generalizable by adding real world training data.

A reasonable next step is to add more and varying simulation data, and then move towards real world data for training. The problem with sparse point cloud remains an issue, which we plan to solve using various processes. Because the main problem is to be sure that selected point to calculate the distance is actually a point on human body, we can use camera + lidar fusion, segmentation and torso detection techniques for verification that selected point is actually part of human body. Another approach would be to use radio beacons. These emit radio signals that can be detected using directional antennas to get position of humans. Rest of the pipeline will remain the same.

One definite improvement is to add more classes to the dataset.
Another enhancement is to predict motion vectors. This means that also the direction of where human is moving is estimated. This can be very helpful when planning vehicle motion. If the trajectory of human is known it can help plan better and avoid any undesirable outcome. For this some other neural network technique will be required as it is very difficult to predict which direction someone is moving by just using single scene image.

Finally, there are a lot of other neural network architectures and implementations, which performance should be evaluated.

7. Conclusion

In this work we have experimented with data from simulation environment containing snowy forest, and humans in random poses to cover our use case of off-road autonomous driving, and object detection focusing on humans only to ensure the safety of humans around an autonomous vehicle.

We predict human bounding boxes and estimate their distance provided we only have monocular camera image available. We trained YOLOv3 on a non-trivial dataset, forest environment with snow, populated with humans in different poses (11 in total) varying from walking, running, crouching to lying down.

Our results show that a model trained for object detection can be modified to predict distance as well and it behaves similarly well after modification. Training input was modified to expect ground-truth value for distance as well. In addition, prediction layer was modified to estimate distance and loss function was changed to take into account the distance error. All of this was needed to make better predictions. Different loss scales were used to get RMSE values down to 1.97, and also different learning rates to see which combination gets us better results quickly. We achieved 60% lower RMSE when compared to LASSO.

All in all, the proposed approach makes it much more convenient for object detection and distance estimation task as there is a single network involved. It also reduces the estimation time when compared to a solution that required two networks e.g. DisNet. Finally, also a large distance estimation error reduction was achieved.

There are a lot of future work to be done that includes addition more simulation data with different environment, people of different ages groups and skin tones, etc. Real-world data needs to be added to generalize the model better. Also add another model that could predict motion vectors to of humans to help vehicle plan better.

References

[1] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” 08-Apr-2018.

[2] V. Murashov, F. Hearl, and J. Howard, “Working safely with robot workers: Recommendations for the new workplace,” *J. Occup. Environ. Hyg.*, vol. 13, no. 3, pp. D61–71, 2016.

[3] T. S. Combs, L. S. Sandt, M. P. Clamann, and N. C. McDonald, “Automated Vehicles and Pedestrian Safety: Exploring the Promise and Limits of Pedestrian Detection,” *Am. J. Prev. Med.*, vol. 56, no. 1, pp. 1–7, Jan. 2019.

[4] “Home - Milrem,” *Milrem*. [Online]. Available: https://milremrobotics.com/. [Accessed: 15-May-2019].

[5] “THeMIS - Milrem,” *Milrem*. [Online]. Available: https://milremrobotics.com/themis/. [Accessed: 15-May-2019].

[6] Contributors to Wikimedia projects, “Object detection - Wikipedia,” *Wikimedia Foundation, Inc.*, 18-Feb-2008. [Online]. Available: https://en.wikipedia.org/wiki/Object_detection. [Accessed: 16-May-2019].

[7] A. Andreopoulos and J. K. Tsotsos, “50 Years of object recognition: Directions forward,” *Computer Vision and Image Understanding*, vol. 117, no. 8. pp. 827–891, 2013.

[8] N. Dvornik, J. Mairal, and C. Schmid, “Modeling Visual Context Is Key to Augmenting Object Detection Datasets,” *Computer Vision – ECCV 2018*. pp. 375–391, 2018.

[9] D. Kim and R. Dahyot, “Face Components Detection Using SURF Descriptors and SVMs,” *2008 International Machine Vision and Image Processing Conference*. 2008.

[10] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* .

[11] G. Von Zitzewitz, “Survey of neural networks in autonomous driving,” Jul. 2017.

[12] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision*, vol. 115, no. 3. pp. 211–252, 2015.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6. pp. 84–90, 2017.

[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal loss for dense object detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, Jul. 2018.

[15] S. Narkhede, “Understanding Confusion Matrix,” *Towards Data Science*, 09-May-2018. [Online]. Available: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. [Accessed: 20-May-2019].

[16] Contributors to Wikimedia projects, “Precision and recall - Wikipedia,” *Wikimedia Foundation, Inc.*, 21-Nov-2007. [Online]. Available: https://en.wikipedia.org/wiki/Precision_and_recall. [Accessed: 14-May-2019].

[17] W. Van Gansbeke, D. Neven, B. De Brabandere, and L. Van Gool, “Sparse and noisy LiDAR completion with RGB guidance and uncertainty,” 14-Feb-2019.

[18] S. Thrun *et al.*, “Stanley: The Robot That Won the DARPA Grand Challenge,” *Springer Tracts in Advanced Robotics*. pp. 1–43, 2007.

[19] U. Wandinger, “Introduction to Lidar,” *Lidar*. pp. 1–18.

[20] {J Carter *et al.*, “*Lidar101: An Introduction to Lidar Technology, Data, and*

Applications," National Oceanic and Atmospheric Administration (NOAA) Coastal Services Center. 2012, Nov. 2012.
[21] R. C. Gonzalez and R. E. Woods, *Digital Image Processing, Global Edition*. Pearson Higher Education, 2017.
[22] "How LiDAR Technology Enables Autonomous Cars to Operate Safely - Velodyne Lidar," *Velodyne Lidar*, 20-Sep-2018. [Online]. Available: https://velodynelidar.com/newsroom/how-lidar-technology-enables-autonomous-cars-to-operate-safely/. [Accessed: 14-May-2019].
[23] D. Marr and T. Poggio, "A computational theory of human stereo vision," *Proc. R. Soc. Lond. B Biol. Sci.*, vol. 204, no. 1156, pp. 301–328, May 1979.
[24] M. Hariyama, T. Takeuchi, and M. Kameyama, "Reliable stereo matching for highly-safe intelligent vehicles and its VLSI implementation," *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No.00TH8511)*. .
[25] L. Nalpantidis and A. Gasteratos, "Stereo Vision Depth Estimation Methods for Robotic Applications," *Depth Map and 3D Imaging Applications*. pp. 397–417.
[26] J. Cech and R. Sara, "Efficient Sampling of Disparity Space for Fast And Accurate Matching," *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007.
[27] A. Kadambi, A. Bhandari, and R. Raskar, "3D Depth Cameras in Vision: Benefits and Limitations of the Hardware," in *Computer Vision and Machine Learning with RGB-D Sensors*, 2014, pp. 3–26.
[28] Y. He, L. Chen, and M. Li, "Sparse depth map upsampling with RGB image and anisotropic diffusion tensor," *2015 IEEE Intelligent Vehicles Symposium (IV)*. 2015.
[29] M. Hansard, S. Lee, O. Choi, and R. P. Horaud, *Time-of-Flight Cameras: Principles, Methods and Applications*. Springer Science & Business Media, 2012.
[30] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised Learning of Depth and Ego-Motion from Monocular Video Using 3D Geometric Constraints," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018.
[31] "RMSE: Root Mean Square Error - Statistics How To," *Statistics How To*, 25-Oct-2016. [Online]. Available: https://www.statisticshowto.datasciencecentral.com/rmse/. [Accessed: 16-May-2019].
[32] B. Marr, "What Are Artificial Neural Networks - A Simple Explanation For Absolutely Anyone," *Forbes*, 24-Sep-2018. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2018/09/24/what-are-artificial-neural-networks-a-simple-explanation-for-absolutely-anyone/. [Accessed: 13-May-2019].
[33] Contributors to Wikimedia projects, "Machine learning - Wikipedia," *Wikimedia Foundation, Inc.*, 25-May-2003. [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning. [Accessed: 16-May-2019].
[34] "Machine Learning Glossary | Google Developers," *Google Developers*. [Online]. Available: https://developers.google.com/machine-learning/glossary/. [Accessed: 19-May-2019].
[35] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10. p. 78, 2012.
[36] Cdiscount Data Science, "A brief overview of Automatic Machine Learning solutions (AutoML)," *Hacker Noon*, 25-May-2018. [Online]. Available: https://hackernoon.com/a-brief-overview-of-automatic-machine-learning-solutions-automl-2826c7807a2a. [Accessed: 15-May-2019].
[37] "CS231n Convolutional Neural Networks for Visual Recognition." [Online]. Available: http://cs231n.github.io/neural-networks-1/. [Accessed: 06-May-2019].

[38] “What is convolutional neural network? - Definition from WhatIs.com,” *SearchEnterpriseAI*. [Online]. Available: https://searchenterpriseai.techtarget.com/definition/convolutional-neural-network. [Accessed: 18-May-2019].
[39] A. S. Walia, “Types of Optimization Algorithms used in Neural Networks and Ways to Optimize Gradient Descent,” *Towards Data Science*, 10-Jun-2017. [Online]. Available: https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f. [Accessed: 18-May-2019].
[40] S. Sharma, “Epoch vs Batch Size vs Iterations,” *Towards Data Science*, 23-Sep-2017. [Online]. Available: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9. [Accessed: 16-May-2019].
[41] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
[42] W. Liu *et al.*, “SSD: Single Shot MultiBox Detector,” *Computer Vision – ECCV 2016*. pp. 21–37, 2016.
[43] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
[44] R. Girshick, “Fast R-CNN,” 30-Apr-2015.
[45] J. Redmon, “Darknet: Open Source Neural Networks in C.” [Online]. Available: https://pjreddie.com/darknet/. [Accessed: 06-May-2019].
[46] “About CUDA,” *NVIDIA Developer*, 06-Mar-2014. [Online]. Available: https://developer.nvidia.com/about-cuda. [Accessed: 17-May-2019].
[47] M. A. Haseeb, D. Ristić-Durrant, A. Gräser, “Long-range obstacle detection from a monocular camera,” *CSCS18*, Sep. 2018.
[48] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, “University of Michigan North Campus long-term vision and lidar dataset,” *The International Journal of Robotics Research*, vol. 35, no. 9. pp. 1023–1035, 2016.
[49] “Nutikas UGV - Milrem,” *Milrem*. [Online]. Available: https://milremrobotics.com/nutikas-ugv/. [Accessed: 16-May-2019].
[50] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles,” *Field and Service Robotics*. pp. 621–635, 2018.
[51] “Unreal Engine | What is Unreal Engine 4.” [Online]. Available: https://www.unrealengine.com/en-US/what-is-unreal-engine-4. [Accessed: 13-May-2019].
[52] AlexeyAB, “AlexeyAB/darknet,” *GitHub*. [Online]. Available: https://github.com/AlexeyAB/darknet. [Accessed: 06-May-2019].
[53] D. G. Kirkpatrick and R. Seidel, “The Ultimate Planar Convex Hull Algorithm?,” *SIAM Journal on Computing*, vol. 15, no. 1. pp. 287–299, 1986.
[54] ManivannanMurugavel, “ManivannanMurugavel/YOLO-Annotation-Tool,” *GitHub*. [Online]. Available: https://github.com/ManivannanMurugavel/YOLO-Annotation-Tool. [Accessed: 06-May-2019].
[55] S. Jadon, “Introduction to Different Activation Functions for Deep Learning,” *Medium*, 16-Mar-2018. [Online]. Available: https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092. [Accessed: 19-May-2019].

[56] “COCO - Common Objects in Context.” [Online]. Available: http://cocodataset.org/#detection-eval. [Accessed: 06-May-2019].

Non-exclusive license to reproduce thesis and make thesis public

I, Asif Sattar

1. herewith grant the University of Tartu a free permit (non-exclusive license) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

Human detection and distance estimation with monocular camera using YOLOv3 neural network,
Supervised by **Karl Kruusamäe** and **Lauri Tammeveski**

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Asif Sattar
20/05/2019